REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
1/94	Scientific Paper

4. TITLE AND SUBTITLE

Incorporating Geometric Constraints into Rule-Based Systems Using Nonlinear Optimization

5. FUNDING NUMBERS

6. AUTHOR(S)

Jo Ann Parikh, and Anne Werkheiser

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

U.S. Army Topographic Engineering Center
ATTN: CETEC-PAO
7701 Telegraph Road
Alexandria, VA 22310-3864

8. PERFORMING ORGANIZATION REPORT NUMBER

R-204

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release;
distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

A transformation procedure is presented for converting rule-based systems with nonlinear constraints into nonlinear optimization problems. The transformation procedure is then applied to an illustrative site identification system. Nonlinear optimisation problems are constructed both for the subset of rules consisting only of logical predicates and for the complete system. Experimental results obtained by solving the nonlinear optimization problems for the site identification system yield a complete description of valid variable bindings for all rules in the system.

This procedure was developed in response to the need to solve logical deployment problems for the U.S. Army. In this application domain, numerical, geometric, and geographic constraints must be incorporated with logical constraints in a uniform framework as in human inference. Rules in the system must be able to express various types of relationships, including relationships in the form of nonlinear constraints. Modeling rule-based systems as nonlinear optimimization problems provides a powerful, uniform framework with the flexibility to handle mixed data types and numerical and geometric relationships.

14. SUBJECT TERMS	15. NUMBER OF PAGES
rule-based systems, nonlinear constraints, site identification, logical deployment problems, geographic constraints	7
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
unclassified	unclassified	unclassified	

Incorporating Geometric Constraints into Rule-Based Systems Using Nonlinear Optimization*

Jo Ann Parikh
Computer Science Department
Southern Connecticut State University
501 Crescent Street
New Haven, CT 06515
E-mail : parikh@scsu.ctstateu.edu

Anne Werkheiser
U. S. Army Topographic Engineering Center
Fort Belvoir, VA 22060-5546
E-mail: anne@pooh.tec.army.mil

ABSTRACT

A transformation procedure is presented for converting rule-based systems with nonlinear constraints into nonlinear optimization problems. The transformation procedure is then applied to an illustrative site identification system. Nonlinear optimisation problems are constructed both for the subset of rules consisting only of logical predicates and for the complete system. Experimental results obtained by solving the nonlinear optimization problems for the site identification system yield a complete description of valid variable bindings for all rules in the system.

This procedure was developed in response to the need to solve logical deployment problems for the U.S. Army. In this application domain, numerical, geometric, and geographic constraints must be incorporated with logical constraints in a uniform framework as in human inference. Rules in the system must be able to express various types of relationships, including relationships in the form of nonlinear constraints. Modeling rule-based systems as nonlinear optimimization problems provides a powerful, uniform framework with the flexibility to handle mixed data types and numerical and geometric relationships.

1. INTRODUCTION

A potentially useful way of organizing the massive volume of information available from multiple sensors with human expertise and decision-making criteria is through facts and rules that can be accessed by expert systems. In these systems numerical, geographical, and geometric information must be combined uniformly with logical information as in human inference to answer queries and solve problems. The rules in the system must be able to express various types of relationships, including constraint relationships containing variables which must satisfy both logical and numerical relationships.

*This work was supported in part by the National Science Foundation under grant IRI-9108638

Previous work in this area includes the development of the class of Constraint Logic Programming (CLP) languages, due to Jaffar and Lassez.[1] CLP(R), a constraint programming language over the domain of real numbers, solves linear constraints by generalizing unification and solves nonlinear constraints by delaying the solution until a sufficient number of variables have been solved to reduce the nonlinear constraints to linear constraints. In CLP(M), a constraint programming language for solving optimization problems, the "ConstrainedMin" and "ConstrainedMax" routines in *Mathematica* are used to solve embedded optimization problems.

A mixed integer linear programming methodology, in contrast to unification and resolution, was developed by Bell *et al.* [2] to perform deduction (compute stable models) at compile-time instead of run-time. Using this approach, run-time query executions can be performed by traditional database "select/project/join" operations. The authors provide experimental results demonstrating the efficiency of the approach compared to standard Prologs. They also point out that the existence of incremental algorithms for solving linear programs suggests that the linear programming methodology for compile-time deduction may be very efficient when performing system updates. Other advantages of this methodology mentioned by the authors are that the ordering of rules and antecedents in the system will not affect the results and that queries will always terminate.

In this paper, we concentrate on performing deduction in rule-based systems using a nonlinear optimization nethodology. The systems under consideration contain rules which require the satisfaction of geometric constraints in conjunction with logical constraints. Facts and rules in the systems describe terrain characteristics and terrain reasoning coupled with decision-making criteria.

A transformation procedure is developed for mapping the rule-based system with geometric constraints into a nonlinear optimization problem which can be solved either at compile-time or at run-time. Domain information, facts, rules, and couplings between logical predicates and geometric constraints are uniformly represented as nonlinear constraints which are positive for invalid solutions (deductions) and zero for valid solutions. The procedure is described for an illustrative site identification system and experimental results of solving the corresponding nonlinear optimization problem are presented.

The remainder of the paper is organized as follows. Section 2 describes the transformation procedure and introduces the illustrative site identification system. Section 3 discusses experimental results obtained by solving nonlinear optimization problems associated with the site identification system. Conclusions and recommendations for future research are outlined in Section 4.

2. TRANSFORMATION PROCEDURE

The goal of the transformation procedure is to map deductive systems with geometric constraints into combinatorial optimization problems. Combinatorial optimization problems can be defined as the class of problems which deal with maximizing or minimizing a function of many variables subject to inequality and equality constraints and integrality restrictions on some or all of the variables. We will be concerned with the equality-constrained optimization problem

$$\begin{aligned} &\text{minimize} && f(x), \\ &\text{subject to} && h(x) = 0, \end{aligned} \tag{1}$$

where $f : \mathbf{R}^n \rightarrow \mathbf{R}$, $h : \mathbf{R}^n \rightarrow \mathbf{R}^m$ and f and g are nonlinear functions.

Variable-free rules are converted into equivalences by noting that if there are two or more rules with the same consequent, they can be combined as below, i.e.

$$\begin{array}{c} B_{1,1} \text{ and } B_{1,2} \text{ and } \cdots \text{ and } B_{1,n} \rightarrow A \\ \cdots \\ B_{k,1} \text{ and } B_{k,2} \text{ and } \cdots \text{ and } B_{k,n} \rightarrow A \end{array}$$

can be rewritten as

$$(B_{1,1} \text{ and } B_{1,2} \text{ and } \cdots \text{ and } B_{1,n}) \text{ or } \cdots \text{ or } (B_{k,1} \text{ and } B_{k,2} \text{ and } \cdots \text{ and } B_{k,n}) \leftrightarrow A$$

In essence, this step of the transformation procedure computes the completion of the ground version of a logic program as outlined in Bell *et al.*[2]

For the site identification system in Figure 1 below, the results of the first step of the transformation for the rules *existing_construction*(a,b), *possible_building_site*(a,b), and *possible_restaurant_site*(a,b) are respectively

$$\textit{existing\_construction}(a,b) \leftrightarrow \textit{chemical\_plant}(a,b) \text{ or } \textit{house}(a,b) \text{ or } \textit{interstate}(a,b). \quad (2)$$

$$\textit{possible\_building\_site}(a,b) \leftrightarrow \textit{flat\_terrain}(a,b) \text{ and (not } \textit{existing\_construction}(a,b)). \quad (3)$$

$$\begin{array}{rcl} \textit{possible\_restaurant\_site}(X,Y) & \leftrightarrow & \textit{possible\_building\_site}(X,Y) \text{ and } X > 3 \text{ and } Y > 3 \\ & & \text{and (distance from } (X,Y) \text{ to } (5,5) < 1.2) \end{array} \quad (4)$$

where a and b are constants obtained by substituting one of the values $1,2,3,4,5$ for X and Y.

The next step in the procedure is to construct constraints on solutions to the nonlinear optimization problem to represent the facts in a rule-based system. Each fact and also each logical predicate appearing in the variable-free form of the system is associated with a real variable in the range $[0,1]$ with 0 denoting a "true" predicate and 1 a "false" predicate. For each fact, f, a constraint of the form $F = 0$ is constructed (where F is the variable associated with the fact f).

Domain constraints are also imposed on the values of the associated variables. In order to ensure that the value of each associated variable will be either "true" or "false", constraints of the form $F(1-F) = 0$ must be included for each variable associated with a logical predicate. For the site identification system (with the exclusion of the last rule), there are a total of 175 domain constraints. Each of the seven predicates in the program, i.e.*chemical_plant*(X,Y), *flat_terrain*(X,Y), *house*(X,Y), *interstate*(X,Y), *lake*(X,Y), *existing_construction*(X,Y), and *possible_building_site*(X,Y), expands into 25 predicates in the variable-free version of the system when the values 1 through 5 are substituted for X and Y.

Logic constraints are represented by equating to zero functions which are non-zero for invalid combinations of truth values and zero for valid sets of truth values. The functions constructed to enforce the logic constraints were formed by

1. enumerating all invalid combinations of truth values for rules which had been transformed into equivalence relations,

2. associating a truth value of "true" to a term of the form $(1 - v)$ and a truth value of "false" to a term of the form v, and

3. multiplying together, for each invalid combination of truth values, the associated terms.

Facts for Site Identification System:

chemical_plant(2, 2).

flat_terrain(1, 1).	*flat_terrain*(1, 2).	*flat_terrain*(1, 3).	*flat_terrain*(2, 1).	*flat_terrain*(2, 2).
flat_terrain(2, 3).	*flat_terrain*(2, 4).	*flat_terrain*(3, 1).	*flat_terrain*(3, 2).	*flat_terrain*(3, 3).
flat_terrain(4, 1).	*flat_terrain*(4, 2).	*flat_terrain*(4, 4).	*flat_terrain*(4, 5).	*flat_terrain*(5, 2).
flat_terrain(5, 3).	*flat_terrain*(5, 4).	*flat_terrain*(5, 5).		

house(1, 4). *house*(1, 5). *house*(2, 5).

interstate(5, 5).

lake(5, 1).

Rules for Site Identification System

existing_construction(X, Y) ← *chemical_plant*(X, Y).
existing_construction(X, Y) ← *house*(X, Y).
existing_construction(X, Y) ← *interstate*(X, Y).

possible_building_site(X, Y) ← *flat_terrain*(X, Y) and (not *existing_construction*(X, Y)).

possible_restaurant_site(X, Y) ← *possible_building_site*(X, Y) and $X > 3$ and $Y > 3$ and (distance from (X, Y) to $(5, 5) < 1.2$)

Figure 1: Site Identification System

This general procedure of constructing a system of nonlinear constraints to represent logic constraints is illustrated below for the rule given in equation 2. The following notation will be used. Associated with predicates *existing_construction*(a, b), *chemical_plant*(a, b), *house*(a, b), *interstate*(a, b), *possible_building_site*(a, b), and *flat_terrain*(a, b) are the variables $E_{a,b}$, $C_{a,b}$, $H_{a,b}$, $I_{a,b}$, $P_{a,b}$, and $F_{a,b}$, respectively. Note that for equation 2, the invalid sets of truth combinations for $(E_{a,b}, C_{a,b}, H_{a,b}, I_{a,b})$ are {t,f,f,f}, {f,t,t,t}, {f,t,t,f}, {f,t,f,t}, {f,t,f,f}, {f,f,t,t}, {f,f,t,f}, and {f,f,f,t} where "t" denotes "true" and "f" denotes "false". The corresponding constraints for *existing_construction*(a, b) are given below:

Logic Constraints for existing_construction(a,b)

$$
\begin{aligned}
(1 - E_{a,b})C_{a,b}H_{a,b}I_{a,b} &= 0 \\
E_{a,b}(1 - C_{a,b})(1 - H_{a,b})(1 - I_{a,b}) &= 0 \\
E_{a,b}(1 - C_{a,b})(1 - H_{a,b})I_{a,b} &= 0 \\
E_{a,b}(1 - C_{a,b})H_{a,b}(1 - I_{a,b}) &= 0 \\
E_{a,b}(1 - C_{a,b})H_{a,b}I_{a,b} &= 0 \\
E_{a,b}C_{a,b}(1 - H_{a,b})(1 - I_{a,b}) &= 0 \\
E_{a,b}C_{a,b}(1 - H_{a,b})I_{a,b} &= 0 \\
E_{a,b}C_{a,b}H_{a,b}(1 - I_{a,b}) &= 0
\end{aligned}
$$

Numeric and geometric constraints embedded in rules are treated analogously to logical relationships. During the process of substitution of values for variables in the first stage of the procedure, several simplifications may be directly applied. For example, in the site identification system, possible sites considered for a restaurant may be limited to those sites where X and Y are both greater than 3. This means that there are possibly four sites, sites $(4,4)$, $(4,5)$, $(5,4)$, and $(5,5)$ which need to be considered in the formulation of the optimisation problem. All possible truth combinations for the rule for *possible_restaurant_site* were then considered. The inequality constraint that the distance between a possible restaurant site and the site $(5,5)$ where the interstate was located must be less than 1.2 was turned into an equality constraint by adding a slack variable. Thus, the distance condition inequality $d((a,b),(5,5)) < 1.2$ became a distance condition equality of the form

$$d((a,b),(5,5)) - 1.2 + slack(a,b) = 0$$

where $slack(a,b)$ is the slack variable associated with $d((a,b),(5,5))$. Let us denote the square of the expression on the left-hand side of the equality in equation 5 above as $Dist^2_{a,b}$. Whenever, in one of the invalid sets of truth combinations, the truth value for the distance condition evaluated to "false", $Dist^2_{a,b}$ was inserted into the associated optimisation constraint product. When the truth value for the distance condition was "true", a desirable function for the associated term in the product would have the property of being close to zero when the distance condition was not satisfied and non-zero if the distance condition was satisfied. A function which we have used in our experiments is $1/(1 + exp(Dist^2_{a,b}))$. Using this function, the logic constraints for *possible_restaurant_site* were

Logic Constraints for possible_restaurant_site

$$
\begin{aligned}
(1 - R_{a,b})P_{a,b} &= 0 \\
(1 - R_{a,b})(1 - P_{a,b})Dist^2_{a,b} &= 0 \\
R_{a,b}(1 - P_{a,b})(1/(1 + exp(Dist^2_{a,b}))) &= 0
\end{aligned}
$$

where $R_{a,b}$ and $P_{a,b}$ are the variables associated respectively with *possible_restaurant_site* and *possible_building_site*(a,b).

If the sample logic program had consisted only of logical information, the objective function to be minimized would have been simply the negative of the sum of the logical variables. Minimising the negative of the sum of variables that assume the value 1 if false (and 0 if true) means that unless explicitly told by the facts or the rules in the logic program that a particular ground atom (assumption) is true, one is to assume that it is false. There is no guarantee that there is a unique optimal solution to this problem or that all solutions are optimal solutions. For further details, reference can be made to the discussion on computation of minimal models in Bell *et al.*[2]

When the logic program consists of geometric as well as logical information, the form of the objective function is more complex. For the site identification system, the objective function $f(\vec{v})$ which was to be minimized consisted of the negative of the sum of 179 variables added together with the sum of four variables of the form $Dist^2_{a,b}$.

3. EXPERIMENTAL RESULTS

A standard approach to solving a constrained nonlinear optimisation problem is to solve an equivalent unconstrained minimization problem. The unconstrained problem typically involves minimising a function consisting of the sum of the original objective function and a weighted sum of non-negative constraint terms called a penalty function. The penalty parameter (coefficient) is adjusted during the course of the convergence procedure to guarantee convergence to a feasible solution.

Recurrent neural networks with time-varying penalty parameters as described by Wang[3] were used in this study to find solutions for the Site Identification System. The first task in the development of a recurrent neural network for solving an optimization problem is to design an energy function whose minimum represents the solution to the optimization problem. Let us assume that the optimization problem can be formulated as a nonlinear programming problem with a single objective function $f(\vec{v})$ in which the goal is to

$$\text{minimize } f(\vec{v}) \text{ subject to the constraint } p(\vec{v}) = 0$$

An energy function can be defined as

$$E[\vec{v}(t), \lambda(t)] = f(\vec{v}(t)) + \lambda(t)p(\vec{v}(t))$$

where the penalty function $p(\vec{v})$ is assumed to be a non-negative, differentiable function which is equal to zero if and only if $\vec{v}$ is a feasible solution to the optimization problem. The penalty parameter $\lambda(t)$ is assumed to be a positive, monotonically increasing function of t. $\lambda(t)$ is initialized to a low value and increased slowly to avoid numerical instability and overpenalisation. The function which was used for $\lambda(t)$ was

$$\lambda(t) = \sum_{\tau=0}^{t} \frac{\sqrt{\tau}}{p(\vec{v}(\tau))c_\lambda}$$

Experiments were conducted using the neural network approach on the Site Identification System for (1) a nonlinear optimization problem with 175 variables obtained by applying the transformation procedure to the system with the exclusion of the *possible_restaurant_site* rule and

for (2) a nonlinear optimization problem with 182 variables obtained by using the transformation procedure on the full version of the system. Convergence to a correct solution was obtained after only three iterations for the first optimization problem. The twenty-five variables associated with the *existing_construction* rule in equation 2 and the twenty-five variables associated with the *possible_building_site* rule of equation 3 were correctly identified as 0 for "true" and 1 for "false". For the second optimization problem, convergence was obtained in under 10,000 iterations to the correct solution.

4. CONCLUSIONS AND FUTURE DIRECTIONS

This research has provided a design methodology for executing queries to rule-based systems containing both logical and geometric constraints. A particularly innovative aspect of our approach is the encoding of logical information with geometric constraints as a nonlinear optimization problem. The feasibility of the approach has been demonstrated on a small, but realistic example. The conclusion reached is that this method has the potential to compute fast and efficient solutions to logical deployment problems and that it should be the subject of more investigation in order to realize its full potential in real-world applications.

Future research directions include extension to full-fledged hybrid knowledge bases. Methods need to be developed to process queries to logic programs which handle classical and non-monotonic negation, reasoning about time, and reasoning about uncertainty in the context of nonlinear optimization methodology. Finally, more understanding is needed of the complex interactions of the new method as it is embedded in the battlefield situations for which it is designed. Deployment of any one asset on the battlefield is legitimately studied in isolation in order to delineate its precise characteristics and the characteristics of the site it requires. But in practice it may be necessary to optimize everything simultaneously in order to express the real situation where the various battlefield assets are competing for prime locations.

BIBLIOGRAPHY

1. J. Jaffar and J.-L. Lassez. "Constraint logic programming", *Proc. 1986 ACM Symp. on Principles of Programming Languages.*

2. C. Bell, A. Nerode, R. Ng, and V. S. Subrahmanian, "Computation and implementation of non-monotonic deductive databases", University of Maryland Technical Report CS-TR-2801 (1991).

3. J. J. Wang, "On the asymptotic properties of recurrent neural networks for optimization", *International Journal of Pattern Recognition and Artificial Intelligence* 5, 4(1991), 581-601.